2011 21st International Conference on Field Programmable Logic and Applications

# Run-time Scalable Architecture for Deblocking Filtering in H.264/AVC-SVC Video Codecs

A. Otero, E. De La Torre and T. Riesgo
Centro de Electrónica Industrial
Universidad Politécnica de Madrid
Spain
{joseandres.otero,eduardo.delatorre,
teresa.riesgo}@upm.es

T. Cervero, S. López, G. Callicó, and R. Sarmiento
Instituto Universitario de Microelectrónica Aplicada
Universidad de Las Palmas de Gran Canaria
Spain
{tcervero, seblopez, gustavo, roberto}@iuma.ulpgc.es

*Abstract*— **Systems relying on fixed hardware components with a static level of parallelism can suffer from an underuse of logical resources, since they have to be designed for the worst-case scenario. This problem is especially important in video applications due to the emergence of new flexible standards, like Scalable Video Coding (SVC), which offer several levels of scalability.**

**In this paper, Dynamic and Partial Reconfiguration (DPR) of modern FPGAs is used to achieve run-time variable parallelism, by using scalable architectures where the size can be adapted at run-time. Based on this proposal, a scalable Deblocking Filter core (DF), compliant with the H.264/AVC and SVC standards has been designed. This scalable DF allows run-time addition or removal of computational units working in parallel. Scalability is offered together with a scalable parallelization strategy at the macroblock (MB) level, such that when the size of the architecture changes, MB filtering order is modified accordingly.**

Keywords— Reconfigurable architectures, scalable architecture, Deblocking Filter, H.264/AVC, SVC

## I. INTRODUCTION

Current multimedia applications demand flexible and high performance systems in order to be implemented. The typical solution is to design hardware-software partitions, where the most computational intensive tasks are done in hardware and the rest of tasks are done in software. It is also necessary to adjust the performance of each hardware component of the system, by selecting the optimal combination of design parameters, like pipelining depth or parallelism level that optimizes the whole system behavior. However, achieving an optimal design remains a challenge, since most real task workloads are dependent on run-time system conditions, such as the application working point, and the input data to be processed [10]. Accordingly, traditional design techniques opt to oversize all the blocks for the worst-case scenario. In all but the worst case, hardware logic will be underused. Conversely, in our approach, a flexible and scalable scheme is explored.

In this paper, the Dynamic and Partial Reconfiguration (DPR) of commercial Xilinx FPGAs is leveraged [1][2]. DPR allows the design of IP cores offering run-time adaptable parallelism, and achieving a flexible assignment of resources. In order to take advantage of this, the proposed approach is based on scalable architectures that can change their size at run-time, following a 2D highly modular and regular array structure template previously proposed in [3]. Hence, the scaling process can be carried out adding or removing regular blocks belonging to the architecture. This feature drastically reduces reconfiguration time and partial bitstream storage needs. Specifically, the explored architecture consists of an arrangement of special purpose Functional Units (FUs) working in parallel. In addition, the regularity of communication patterns through the array reduces the number of distributed memory accesses.

Current video standards allow different levels of scalability and profiles [4]. However, static parallelization approaches have to be designed to support the worst cases since they are not capable of adapting their behavior to the variable system demands. Furthermore, there exists an interest from both the Joint Video Team of the ITU and ISO/IEC standardization organizations, in developing scalable media formats to come up with the Scalable Video Coding (SVC) standard, an extension of H.264/AVC [5][6]. An SVC video bitstream might contain all the hierarchical information needed for the three supported scalabilities (quality, spatial and temporal resolutions). However, this bitstream can be decoded in several devices simply by considering different amount of information. Considering the large number of combinations in the allowed SVC scalability levels, flexible mechanisms are required to adapt the characteristics of the decoder to the requirements of each client during run-time. The use of the run-time reconfigurable scalable architecture proposed in this paper allows for an adaptable scenario. It might be composed of different blocks in charge of video decoding tasks, with the capability of dynamically adapting its size, and accordingly, its performance, to the type and levels of scalability selected by the users. At the same time, this architecture is able to balance, at run-time, the area-performance-energy consumption of each block. This scheme maximizes the use of reconfigurable resources.

Going further in the profiling of the SVC decoding process, the Deblocking Filter is not only one of the most computationally intensive tasks of the standard, but it is also highly dependent on the selected video profile [7]. This is the motivation for the analysis and implementation of a scalable Deblocking Filter fully compliant with H.264/AVC and SVC standards, following the scalable architecture approach. In addition, a parallelization scheme is proposed that performs the scaling process consistent with the data locality restriction desired in the architecture

978-0-7695-4529-5/11 $26.00 © 2011 IEEE
DOI 10.1109/FPL.2011.72

The rest of this paper is organized as follows. In Section 2, the role of the Deblocking Filter in H.264 and SVC, focusing on the importance of flexibility, is described. Then, a review of the state of the art on parallel architectures is shown in Section 3. In Section 4, the system architecture is detailed, including the parallelization strategy. Section 5 goes through specific design issues regarding run-time scalability, while in Section 6 some implementation details and results are shown. Finally, in Section 7 some conclusions and future work are discussed.

## II. DEBLOCKING FILTER

Recent video coding standards like H.264/AVC and SVC provide high performance and high grade of flexibility in comparison with their predecessors, but at the cost of a higher level of complexity and computational cost. Due to their constraints and complexity, it is unfeasible to ensure real-time restrictions using exclusively software devices. In order to solve this drawback, current developments combine hardware/software embedded solutions by moving computationally intensive functions to hardware (HW); whereas the remaining tasks are implemented in software (SW). Thanks to profiling results, such as [20], [21], [22] and [23]; it is possible to decide which functions deal with HW or SW characteristics. Within the encoding and decoding processes, more than 30% of the whole computation time is due to the deblocking filter (DF). In this sense, the DF stands out like a perfect candidate for being implemented in hardware.

DF reduces the visual perfection of blocking artifacts introduced by other encoding or decoding operations, an effect which is particularly characteristic with block-based coding standards. The inclusion of DF in the loop reduces the bit rate by 5%-10 %, while retaining the same visual quality [13]. A detailed description of the DF algorithm can be found in [14]. According to the H.264/AVC and SVC standards, the filtering operation is performed MB-wise. Each MB is composed of 16 basic blocks, each one comprising of an arrangement of 4 x 4 pixels. As depicted in Figure 1, vertical edge V0 of the block is conventionally filtered horizontally first from top to bottom, followed by edge V1, edge V2, and edge V3. Vertical filtering is performed in a similar way.

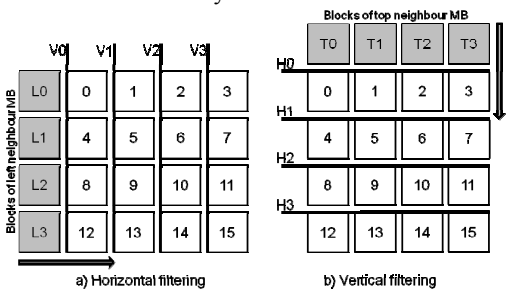


Figure 1. MB edges for a) horizontal and b) vertical filtering

The edge H0 is vertically filtered from the left to the right, followed by the edge H1, H2, and H3. For each edge, the filter takes four pixels of each block at both sides of the edge as its inputs. Consequently for filtering one MB there exist direct dependencies with its upper and left neighbors, which are located in different MBs, so they have to be filtered first. In addition, the standard imposes precedence on horizontal filtering over vertical. The specific filtering algorithm applied to each MB, as well as its strength, depends on the coding mode of each MB, some characteristics of the video sequence, as well as some thresholds at pixel level.

## III. RELATED WORK

Adapting parallelism's levels of processing cores at run-time has received researchers' attention from different perspectives, most of them taking advantage of the unique computation-on-demand possibilities offered by FPGAs including DPR features. For example, some existing approaches explore scheduling and optimal parallelism selection issues, rather than considering architectural challenges. For instance, PARLGRAN [9] is a framework that deals with mapping and scheduling aspects derived from using dynamic parallelism selection. Its purpose is to maximize the performance of an application task chain by selecting the optimal parallelism granularity for each application task. The variable levels of granularity are achieved by instantiating several copies of each task implementation working concurrently. As a consequence, the task workload is equally divided among all instances, reducing proportionally the total execution time. The main limitation of this work is its restricted applicability to data-parallel tasks. Thus, it only considers tasks without dependences between disjoint data blocks, an assumption that is not valid for DF and many other algorithms. In [10], a similar problem is tackled, but this case considers dynamically adaptive reconfigurable accelerators. They propose balancing area and execution time while unrolling application loops, dependant on the inputs and tasks running concurrently in the reconfigurable area. The more the loop is unrolled, the higher the parallelism. Since this work is focused on accelerator selection, no architectural novelties on how different accelerators are built are provided.

After having reviewed existing DPR state-of-the-art approaches working under dynamic conditions, some proposals regarding dynamically scalable architectures will be analyzed next. Achieving scalability by means of DPR allows freely reusing configurable areas for other cores within the reconfigurable logic. However, most existing scalability related works are oriented to adapt core functionality or operation quality, rather than to set area-performance tradeoffs. For instance, the scalable FIR filter provided in [11], offers the capability of adapting the number of taps to adjust the filter order, offering a compromise between filtering quality and required resources. Furthermore, a scalable DCT implementation in [12] allows varying the number of DCT coefficients that are calculated by the core, in order to adapt the number of terms that will be subsequently quantified, and therefore adjust video coding quality. Also a scalable Deblocking filter architecture is proposed in [7], utilizing both DPR and the H.264/AVC deblocking filter. However, this work explores block level parallelism, where the variation of the number of processing units working in parallel affects the execution time of a single macroblock (MB), without dealing with MB dependences. This approach, while still being interesting for flexibility purposes, is limited by the maximum number of 4×4 blocks that can be processed

simultaneously, in one MB, with no further scalability levels. In addition, the designed floorplanning does not allow for easy reuse of the area released when shrinking the filter, so real area-performance tradeoffs cannot be easily achieved.

Compared with previously reported architectures, in this work a dynamically scalable Deblocking filter is proposed in order to be integrated within run-time flexible decoders. It exploits a coarser granularity, compared with state of the art approaches, and it allows reusing the released area, in order to balance area-performance tradeoffs. Scalable DF follows a parallelization strategy previously proposed in [8], fully compatible with the scaling process of the architecture, incorporating data dependences between MBs, and allowing scalability at any level, from just one functional unit (FU) up to the maximum number of resources available in the reconfigurable logic area.

## IV. Global System Architecture

In this section, both the proposed algorithm parallelization as well as the global architecture are described focusing on general design issues, while reconfiguration details are offered in next subsection.

### A. Proposed Algorithm Parallelization Strategy

The number of elements working in parallel in the proposed scalable architecture can be modified at run–time by means of DPR. Larger architectures will be able to process more MBs at a time; however, in order to obtain consistent results, the parallelization strategy has to be scalable itself, since it must respect the data dependences between MBs independently of the size of the architecture. In this sense, all MBs are directly dependent on their left and top filtered neighbor MBs. A detailed analysis of dependencies, as depicted in Figure 2, reveals that current MB horizontal filtering depends on previous neighbor MB vertical filtering, and that the current MB vertical filtering depends on top neighbor horizontal filtering. In Figure 2, $MB_H$ refers to a MB after having been filtered horizontally and $MB_{HV}$ after it has been filtered both horizontally and vertically. To finish processing a MB, $MB_{HV}$ has to be filtered horizontally and vertically together with their right and top neighbors, respectively. Once $MB_{HV}$ is filtered horizontally again, it is depicted as $[MB_{HV}]$.

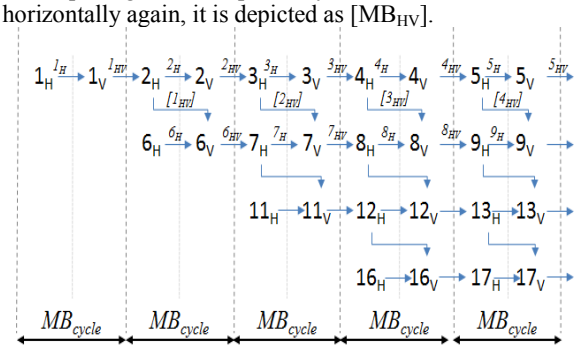


Figure 2. MB data dependencies and scan order proposal for a 6MBs width video frame

A possible solution to exploit MB-level parallelism might entail using a wavefront order in the same way as state-of-the-art multiprocessing solutions [15]. However, this approach needs to wait twice as many clock cycles for filtering a full MB (MBcycle), that is, until $[MB_{HV}]$ is available, before the subsequent core starts processing. Considering this fact, an optimized wavefront order scan has been proposed by the authors in [8], in which horizontal and vertical filtering are separated in sequential stages. As a result, an MB can be filtered one MBcycle earlier, as compared with previous approaches. This is because when the MB vertical filtering begins, the top MB horizontal filtering has already finished, assuming $[MB_{HV}]$ is available.

### B. Proposed DF Architecture

The core of the proposed architecture is a coarse grain homogeneous array of FUs, as depicted in Figure 3. Each unit is able to carry out a complete filtering operation on an MB, such that the full array can process in parallel a region of the image. A more detailed description of each FU can be found in [24]. The main strengths of the proposed structure are its inherent parallelism, regular connections and data processing capabilities. In order to feed each FU with the required MBs, as well as to synchronize the array, modules in charge of controlling input and output FIFO memories have also been included.

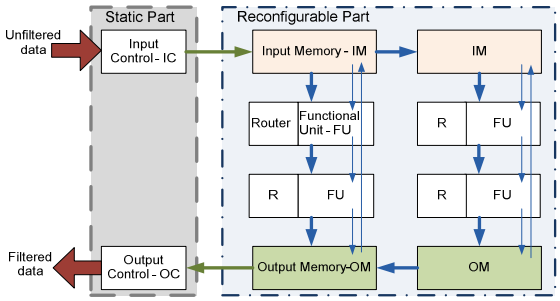


Figure 3. Processing array structure

To respect the processing order defined by the proposed parallelization pattern, a mechanism in charge of the generation of the valid sequence of MB addresses has been included in a hardware module called *Input Controller (IC)*.

This module receives the corresponding MBs sequentially from the memory, and sends them to the array of FUs. In addition, in order to parallelize data provision to the FUs, other modules named *Input Memory* (IM) have been included at the top of each column of the processing array. Main components of these blocks are FIFO memories that distribute the suitable MBs vertically across each column. With the purpose of capturing the MB associates with the corresponding units, a module called *Router* has been attached to each FU. Once all FUs have been fed with the necessary MBs, they are processed in parallel. MB routers are also in charge of transmitting the processed MBs to the elements after the FU, called Output Memories. These blocks are based on FIFO memories that store the elements received from the vertical connection, and transmit them again in sequential order to the Output Controller (OC). This OC will send back the processed MBs to the external memory. Data sending, processing and results transmission stages have been pipelined and overlapped iteratively in two phases, as shown in Table I.

All the blocks of the architecture include distributed control logic to manage data transmission. Distributed control makes the architecture fully scalable, by means of the addition or removal of modules. These modules automatically communicate only with their neighbors using shared signals, without having to implement a centralized control designed *ad hoc* for each possible architecture size, which would reduce scalability.

TABLE I. TASK DISTRIBUTION FOR EACH ELEMENT

| | Elements of the Architecture | | | |
|---|---|---|---|---|
| | ***IM*** | ***Router*** | ***FU*** | ***OM*** |
| **Phase H** | Rx MBt+1 from IC | Tx Filtered MBt-1 | Filtering H MBt | Rx Filtered MBt-1 |
| **Phase V** | Tx MBt+1 to Routers | Rx MBt+1 | Filtering V MBt | Tx Filtered MBt-1 |

### C. MB to FU Allocation Policy

Since horizontal and vertical processes for each MB have to share data corresponding to the full MB, both are carried out in the same FU to minimize communication costs. In addition, according to existing dependencies, semi-filtered data ($MB_{HV}$ and $[MB_{HV}]$ according to the nomenclature in Figure 2) have to be shared between the FU filtering an MB, and the FUs in charge of their top and left neighbors. To reduce this overhead, an MB will always be filtered in the same unit as its left neighbor, and its top neighbor will be filtered in the unit below the MB. As explained in the implementation section, specific connections between FUs have been created to allow the exchange of this semi-filtered information, without involving *routers*. The final allocation sequence for filtering all MBs within a frame is dependent on the total number of FUs of the architectural array and also on the number of MBs of the height image frame. Thus, each FU filters all MBs contained in a particular row of the frame while always respecting data dependencies. However, if the number of FUs is smaller than the height of the MBs in a frame, the filtering process is modified. The frame will be processed by stripes with a height the same as the number of FUs in the array.

This proposal not only reduces the amount of transferred information among FUs, but also, unlike state-of-the-art proposals, only the current MB must be requested from the external memory for each filtering process, as the information related with the neighboring MBs is received during horizontal and vertical filtering stages from other FUs, as explained above. Consequently, data transferred between the external memory and DF is also reduced.

## V. ARCHITECTURAL DESIGN FOR RUN-TIME SCALABILITY

One of the main advantages of using this kind of highly modular and parallel architecture, when considering exploiting DPR, is the straight forward nature of generating the design partitioning following the Modular Design Flow [16]. Thus, each module will be treated as a reconfigurable element on its own. In consequence, their VHDL descriptions are synthesized, mapped, placed and routed independently. However, since most of the components of the architecture are equal, a unique version of each type (IM, OM and FU) will have to be generated, and afterwards it can be reused in different positions of the array In this section, design issues corresponding to each module type, as well as main floorplanning decisions taken to achieve scalability, will be described.

Communications between reconfigurable elements, using bus-macros (BMs) [17], will also be detailed. The latest Xilinx dynamic reconfiguration design flow, from the v12 release onward, avoids the use of these fixed macros to guarantee the correctness of communications across modules frontiers [18]. Instead of BMs, elements called Partition Pins are automatically inserted by the tool in all frontier pins corresponding with all of the reconfigurable module types to grant communication validity. Even though it reduces DPR overhead, this approach does not allow module relocation in different positions of the FPGA, as this is unsuitable for this kind of scalable architecture because of module replication. In the following subsections, the implementation of each basic module of the architecture is described.

### A. Router and FU

Since each FU is always attached to its router, both have been implemented inside a single reconfigurable module. As mentioned before, the role of the router is to capture the first MB received from the IM in each processing stage, and then transmit it without changing the subsequent MBs to the FUs below. Both data and control vertical connections among routers and the IM/OM have been included in the module border, as shown in Figure 4. Additionally, specific point-to-point connections with the FU below have been created to exchange semi-filtered information, as described in the previous section. In the case of the last FU of the column, the next FU is located at the top of the next column. To tackle this issue, a bypass logic was included both in the OM and the FUs blocks to send it upward. In the case of the FU, the bypass connection is highlighted in Figure 4.

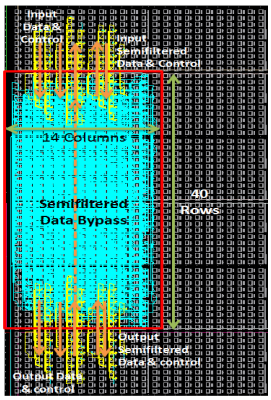


Figure 4. Fig. 4 Router and FU design

North and South connections of the module are completely symmetric, since both use BMs in the same positions. Consequently, FU and Router modules' design can be stacked in vertically aligned positions inside the FPGA. However, it cannot be replicated in positions horizontally, because BRAM columns are located in different positions within each right and left side of a region. To overcome this limitation a different module

compatible with the FPGA area to each side has been created, including the same local behavior, but with a different floorplanning design. Consequently, two different versions of the FU exist. This is shown in Figure 6, representing the full architecture.

### B. Input Memory (IM)

Unfiltered MBs coming from the external video frames' memory are transmitted across the IMs' FIFOs, retaining the MBs to be processed by the column of FUs immediately below it. Consequently, the memory size of the IM limits the maximum vertical size of the architecture. In the final implementation, due to physical restrictions of the FPGA, the architecture was limited to a 2×3 size. Once this memory has been filled, the IM distributes the MBs in the vertical direction, to the FUs of the same column. Consequently, both horizontal and vertical BMs have been included, as is shown in Figure 5.

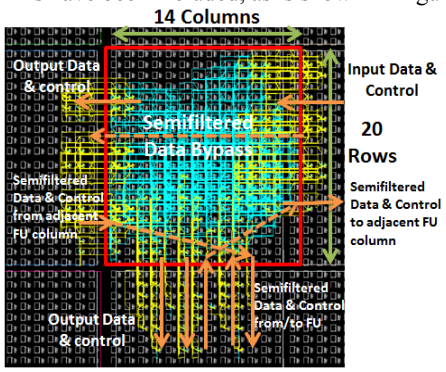


Figure 5. Fig.5 IM floorplanning design

Lines to transmit semifiltered MBs have been included for the worst case scenario, as it was explained in section A. Consequently, the IM receives semifiltered data from the FU bypass output, and send it to the IM on the right. In addition, this module sends semifiltered data from the left IM, to the first unit of the column below. Furthermore, in the case of the last column a special communication module has been implemented to send this data back to the IC. To make this feasible, a special horizontal semifiltered bypass has been included through the IM. To tackle the heterogeneity problem of the FPGA, two versions of this block have been implemented to allow for 2D reconfigurations, as shown in Figure 6.

### C. Output Memory (OM)

This block includes the logic in charge of receiving the completely filtered MBs from the FU column above, as well as transmitting data to the output controller. It also includes inputs and outputs for transmitting semifiltered MBs. Specifically, it bypasses data coming from the semifiltered MB output of the FU immediately above to its semifiltered bypass input, so it can be transmitted to the next column, as has been already been detailed.

### D. Input Controller (IC)

The IC is the input block of the architecture, and it is the communication point with the static part of the system. This block is not dynamically reconfigured when the DF is scaled, only certain registers are configured from the external embedded processor, in order to indicate the current dimensions of the DF and the size of the video frame. With these dimensions, the IC generates the correct MB reading address sequence for any size architecture. Bus-macros corresponding to MB data and control signals have been included, to communicate with the adjacent IM.

### E. Output Controller(OC)

The OC is also part of the static design. It receives data from OMs, and sends it back to the video memory. Consequently, BMs have to be located across the static-reconfigurable borders to allow for different size architectures. However, future work will be carried out to communicate OC outputs with IMs to have a unique communication module with the static area.

The scalability of the full architecture is shown in Figure 6. Because the columns are different, 8 independent bitstreams have been generated, one for the IC, one for the OC, two per each FU, IM and OM. In addition, the communication modules have been included to close open connections. Thus, the maximum size achieved is 2×3, using the right half of a medium size Virtex-5 FPGA (xc5vlx110t).

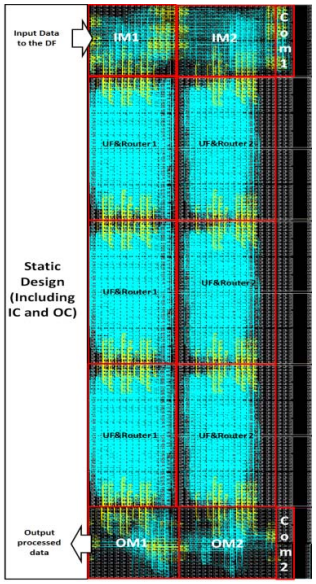


Figure 6. Complete Architecture design

## VI. IMPLEMENTATION DETAILS AND RESULTS

This section collects the implementation results obtained considering both the architecture itself and the DPR issues.

### A. Architectural Details

Within the reconfigurable stage of this architecture, the array might be formed by different number of FUs. In this section, the architecture has been implemented without considering dynamic and partial reconfiguration issues. In other words, different sizes are achieved after a new synthesis process of the full architecture. The number of units varies according to the performance or the on-demand environment constraints. Depending on how the FUs are distributed into the array ($m \times n$), different configurations are obtained, where $m$ and $n$ are the width and height of the array. For a specific amount of FUs there exist several configurations, characterized by having the

same computational performance, but requiring different HW resources and with different data transfer delays. In this scenario, resource utilization of each basic block of the architecture is shown in Table II.

TABLE II. RESOURCES UTILIZATION PER BLOCK

| | Synthesis results using V5LX110T | | | | |
|---|---|---|---|---|---|
| | *IC* | *OC* | *IM* | *OM* | *Router&FU* |
| **Slices reg.** | 357 | 116 | 172 | 124 | 2004 |
| **Slices LUTs** | 444 | 108 | 134 | 226 | 2386 |
| **Block RAM/FIFO (36kb)** | 1 | 0 | 2 | 2 | 8 |

With regards to synthesis results, which are more detailed in [8], the FU limits the maximum operation frequency of the whole architecture at 124 MHz Performance variations are shown in Table III. This data outlines the minimum operation frequency necessary for processing different video formats at 30 frames per second (fps), coinciding with real-time constraint. Each column refers to a specific number of FUs, while each row expresses frequency values for a specific video format ($M \times N$), where $M$ and $N$ are its width and height in MBs respectively.

Using many FUs is not efficient for processing the smallest video formats, since some units will keep idle during filtering execution. The limit of FUs is determined by the height of the video format expressed in MBs. As an example, SQCIF and QCIF formats are 6 and 9 MBs high respectively; as a consequence configurations with more than 6 or 9 FUs are not appropriate for these formats. On the other hand, using a low number of FUs is not possible when processing the highest multimedia format, UHDTV, since its associated configurations need more than 200 MHz for real-time performance, whereas the maximum frequency of this architecture is 124 MHz.

TABLE III. MAXIMUM FREQUENCY FOR REAL-TIME (30FPS)

| **Format @30fps ($M \times N$)** | **Max. Freq. (KHz)** | | | | | |
|---|---|---|---|---|---|---|
| | *1 FU* | *2 FUs* | *3 FUs* | *4 FUs* | *8 FUs* | *16 FUs* |
| **SQCIF (8×6)** | 299 | 156 | 112 | 106 | NA | NA |
| **QCIF (11×9)** | 617 | 343 | 218 | 205 | 137 | NA |
| **CIF (22×18)** | 2,471 | 1,241 | 836 | 692 | 418 | 280 |
| **4CIF (44×36)** | 9,884 | 4,948 | 3,307 | 2,215 | 1,678 | 842 |
| **16 CIF (88×72)** | 39,536 | 19,774 | 13,191 | 9,902 | 4,985 | 2,789 |
| **HDTV 1080 (120×67)** | 50,544 | 25,465 | 17,228 | 12,748 | 6,757 | 3,762 |

### *B. Dynamic Reconfiguration Details*

In this section, details regarding the reconfigurability of the architecture are shown. Adapting the architecture to be dynamically scalable implies the addition of BMs, as well as the design of an independent floorplanning for each module. As previously shown, each module has been designed to occupy extra area, in order to be able to route all internal signals within the reconfigurable region. Thus, even though not all logical resources within the region are occupied, the entire region is necessary to come up with a fully routed design. Consequently, resource utilization is increased, as can be seen in Table IV, regarding each element, and in Table V, concerning to different sizes of the full core. Information in Table IV can be compared with Table II, to see the overhead of designing for DPR. Only main reconfigurable modules have been included.

TABLE IV. RESOURCES IMPACT OF DESIGNING FOR DYNAMIC SCALABILITY

| **Logical Resources** | **Elements of the Architecture** | | |
|---|---|---|---|
| | *IM* | *Router&FU* | *OM* |
| **Slices LUTs** | 2080 | 4160 | 2080 |
| **Slices Registers** | 2080 | 4160 | 2080 |
| **Block RAM/FIFO (36kb)** | 4 | 8 | 4 |

TABLE V. RESOURCES OCCUPANCY OF THE FULL RECONFIGURABLE ARRAY

| **Equivalent resources** | **Size** | | | |
|---|---|---|---|---|
| | *1×2* | *1×3* | *2×1* | *2×2* |
| **Slices LUTs** | 12480 | 16640 | 16640 | 24960 |
| **Slices Registers** | 12480 | 16640 | 16640 | 24960 |
| **Block RAM/FIFO (36kb)** | 24 | 32 | 32 | 48 |

Values for area usage for each element directly impact the size of the corresponding bitstreams, and consequently, the DF reconfiguration time itself. Details about the reconfiguration process are outside the scope of this work. It is carried out by means of an Internal Configuration Access Port (ICAP) controller described in [25], which includes specific features to allow module relocation, specifically addressing these kinds of scalable application. Through the regularity of the architecture, and making use of the DPR, only 8 different bitstream files are enough for configuring any m×n size. These bitstreams can be replicated in different positions of the FPGA to scale the core to any valid size. Consequently, to add an extra row to the DF, only five new modules have to be configured (two corresponding to the new FU row, the output memories and the communication module in the new positions), while the others will remain unchanged. Furthermore, all these bitstreams are already configured in other positions of the FPGA, so that they can be directly replicated from the inside of the configuration memory, without accessing external memories.

## VII. CONCLUSIONS AND FUTURE WORK

This work addresses the design of spatially scalable architectures which are able to adjust their size at run-time, by means of dynamic and partial reconfiguration of commercial FPGAs. Exploiting this feature, a variable parallelism level adjustable to changing application requirements is achieved. Thus, the area-performance trade-off can be balanced at run-time. Furthermore, these benefits are applied to the design of a dynamically scalable Deblocking Filter architecture, where size can be adapted on-line to fulfil the variable requirements of the different profiles and scalability levels of H.264/AVC and SVC video coding standards.

Future work is being carried out in order to improve both the architecture design and its dynamically reconfigurable implementation. Regarding the architecture, performance will be enhanced by optimizing the allocation strategy of MBs to FUs. At the moment, this strategy is fixed. Given a frame size as well as certain DF dimensions,

each FU will always process the same MBs. This strategy simplifies DF control, but a new video frame cannot be processed until the previous one has been completely filtered, introducing an extra overhead. In addition, memory consumption and its distribution within the FU will be also optimized, reducing the area of each FU module. This improvement will also impact results of designing for DPR, since routing inside each module will be simplified. Accordingly, FUs will be floorplanned in narrower regions, looking for the homogeneity of both FU columns.

## ACKNOWLEDGMENTS

This work is supported by the Spanish Ministry of Science and Innovation and European Union (FDER funds) in the context of Dynamic Reconfigurability for Scalability In Multimedia Oriented Networks (DR. SIMON) project, under contract TEC2008-065846-C02.

## REFERENCES

[1] P. Sedcole, B. Blodget, T. Becker, J. Anderson, and P. Lysaght. Modular dynamic reconfiguration in virtex FPGAs. Computers and Digital Techniques, IEE Proceedings -, 153(3):157–164, May 2006

[2] Becker, J.; Hubner, M.; Hettich, G.; Constapel, R.; Eisenmann, J.; Luka, J.; , "Dynamic and Partial FPGA Exploitation," *Proceedings of the IEEE*, vol.95, no.2, pp.438-452, Feb. 2007

[3] Otero, A.; de la Torre, E.; Riesgo, T.; Krasteva, Y.E.; , "Run-Time Scalable Systolic Coprocessors for Flexible Multimedia SoPCs," *International Conference on Field Programmable Logic and Applications (FPL), 2010*, vol., no., pp.70-76, Aug. 31 2010-Sept. 2 2010

[4] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi. Video coding with H.264/AVC: tools, performance, and complexity. IEEE Circuits and Systems Magazine, 4(1):7–28, Feb Jan 2004.

[5] Schwarz, H.; Marpe, D.; Wiegand, T.; , "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.17, no.9, pp.1103-1120, Sept. 2007

[6] ITU-T Rec. H.264 and ISO/IEC 14496-10. *H.264/AVC extension* (Scalable Video Coding - SVC). *Advanced Video Coding for Generic Audiovisual Services*, Version 8: 2007 – Version 10: 2009.

[7] Khraisha, R.; Jooheung Lee; , "A scalable H.264/AVC deblocking filter architecture using dynamic partial reconfiguration," *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), 2010*, pp.1566-1569, 14-19 March 2010

[8] Cervero, T.; Otero, A.;, López, S.; De La Torre, E.; Callicó, G.;,Sarmiento, R.; Riesgo, T. ; ,"A Novel Scalable Deblocking Filter Architecture for H.264/AVC and SVC video codecs*", In the Proceedings of the 2011 IEEE International Conference on Multimedia and Expo*. To be published. (ICME 2011)

[9] Banerjee, S.; Bozorgzadeh, E.; Dutt, N.; , "Exploiting Application Data-Parallelism on Dynamically Reconfigurable Architectures: Placement and Architectural Considerations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* , vol.17, no.2, pp.234-247, Feb. 2009

[10] Like Yan; Yuan Wen; Tianzhou Chen; , "Input-Driven Reconfiguration for Area and Performance Adaption of Reconfigurable Accelerators," *IEEE 13th International Conference on Computational Science and Engineering (CSE), 2010*, vol., no., pp.237-244, 11-13 Dec. 2010

[11] Chang-Seok Choi; Hanho Lee; , "An Reconfigurable FIR Filter Design on a Partial Reconfiguration Platform," *First International Conference on Communications and Electronics, 2006. ICCE '06.*, vol., no., pp.352-355, 10-11 Oct. 2006

[12] Jian Huang; Jooheung Lee; , "A Self-Reconfigurable Platform for Scalable DCT Computation Using Compressed Partial Bitstreams and BlockRAM Prefetching," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.19, no.11, pp.1623-1632, Nov. 2009

[13] Wiegand, T.; Sullivan, G.J.; Bjontegaard, G.; Luthra, A.; , "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology,* , vol.13, no.7, pp.560-576, July 2003

[14] List, P.; Joch, A.; Lainema, J.; Bjontegaard, G.; Karczewicz, M.; , "Adaptive deblocking filter," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.13, no.7, pp.614-619, July 2003

[15] Van der Tol, Erik B.; Jaspers, Egbert G.; "Mapping of H.264 decoding on a multiprocessor architecture", Proceedings of the SPIE, Volume 5022, pp. 707-718 (2003).Image and Video Communications and Processing 2003.

[16] Xilinx Modular Design Flow: www.xilinx.com/itp/xilinx7/books/data/docs/dev/dev0038_8.html

[17] Xilinx Bus-Macro description: www.xilinx.com/itp/xilinx7/books/data/docs/dev/dev0038_8.html#wp1103560

[18] Partial Reconfiguration User Guide V12.1: www.xilinx.com/support/documentation/sw_manuals/xilinx12_1/ug702.pdf

[19] Hübert, H. and Stabernack, B.; "Profiling-Based Hardware/Software Co-Exploration for the design of Video coding architectures"; *Circuits and Systems for Video Technology, IEEE Transactions on,* vol.18, no.11, pp. 1680-1691, 2009.

[20] M. Horowitz, A. Joch, F. Kossentini and A. Hallapuro; "H.264/AVC baseline profile decoder complexity analysis". *Circuit and Systems for Video Technology, IEEE Transactions on;* vol. 13, no7, pp. 704-716, 2003.

[21] I. Werda, T. Dammak, T. Grandpierre, M. Ayed and N. Masmoudi. "Real-time H.264/AVC baseline decoder implementation on TMS320C6416". *Journal of Real-Time Image Processing*. pp.1-18, Springer Berlin, 2010

[22] M. Wien, H. Schwarz and T. Oelbaum, "Performance Analysis of SVC", ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, Doc. JVT-U141, 2006

[23] N.J. Suárez, G.M. Callicó, S. López, J.F. López and R. Sarmiento, "Performance analysis of the scalable video coding (SVC) extension of H.264/AVC for constrained scenarios", Proc. Of SPIE, vol.8067, April 2011(in press)

[24] Cervero, T., Otero A., De la Torre, E., López, S., Callicó G., Riesgo, T. and Sarmiento R; "Scalable 2D architecture for H.264 SVC deblocking filter with reconfiguration capabilities for on-demand adaptation" Proceedings of the SPIE, vol. 8067, April 2011. (in press)

[25] Otero, A.; Morales-Cas, A.; Portilla, J.; de la Torre, E.; Riesgo, T.; , "A Modular Peripheral to Support Self-Reconfiguration in SoCs," *13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD), 2010*, vol., no., pp.88-95, 1-3 Sept. 2010